UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/894,700	06/28/2001	Gopala Krishna R. Kakivaya	MS174297.1	5155

7590 10/04/2004

Himanshu S. Amin
Amin & Turocy, LLP
1900 E. 9th Street, 24th Floor
Cleveland, OH 44114

EXAMINER	
SHAH, NILESH R	

ART UNIT	PAPER NUMBER
2127	

DATE MAILED: 10/04/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/894,700	R. KAKIVAYA ET AL.	
	Examiner	Art Unit	
	Nilesh Shah	2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 June 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *1/16/2002*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1 –34 are presented for examination.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bak et al (5,999,732) (hereinafter Bak) in view of J. Yantchev et al., "Adaptive, low latency, deadlock-free packet routing for networks of processors", IEE Proceedings, 136, 178-186 (May 1989) (hereinafter Yantchev).

4. As per claim 1, Bak teaches the invention substantially as claimed including a system for mitigating problems associated with automatic execution of initialization code, the system comprising:

an initialization method activator adapted to call a class initialization method at a pre-determined execution point (col. 2 lines 17-27, col. 5 lines 25-34).

5. Bak does not specifically teach the use of a deadlock analyzer.

   Yantchev teaches a deadlock analyzer adapted to determining whether running the class initialization method will produce a deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61).

6. It would have been obvious to one skilled in the art at the time of the invention to combine the teachings of Yantchev and Bak because Yantchev's system of looking at deadlocks and trying to prevent them would improve Bak's system by avoiding such network delays.

7. As per claim 2, Bak teaches a system where the initialization method activator is further adapted to check whether a class is initialized and, if the class is not initialized, to call the class initialization method (col. 4 lines 17-23, col. 8lines 22-30).

8. As per claim 3, Yantchev teaches a deadlock analyzer is further adapted to determine whether calling the class initialization method will generate a deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

9. As per claim 4, Bak teaches system where the predetermined execution point is at least one of one of a caller's just in time compilation time, a callee's just in time compilation time, an initial field access time, an initial method access time, an

initial static field access time and a first access of pre- compiled code where no just in time compilation occurs (col. 8 lines 22-30, col. 9 lines 35- 46).

10. As per claim 5, Bak teaches system where the initialization method detector is further adapted to associate initialization check code with one or more components associated with a runtime environment, where the initialization check code is operable to determined whether a class has been initialized (col. 8 lines 22-30, col. 9 lines 35- 46).

11. As per claim 6, Yantchev teaches system where the initialization check code is further operable to determine whether calling the class initialization method will generate a deadlock, and if a deadlock will be generated, to resolve the deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

12. As per claim 7, Bak teaches system where the initialization check code is run at one of a caller's just in time compilation time, a callee's just in time compilation time, a first field access time, a first method access time, a first static field access time and a first access of pre-compiled code where no just in time compilation occurs (fig. 7a element 503, col. 8 lines 1-9).

13. As per claim 8, Yantchev teaches system where the deadlock analyzer analyzes a wait for graph (page 179 col. 2 lines 26-30).

14. As per claim 9, Yantchev teaches system where the deadlock analyzer is further adapted to resolve a deadlock associated with running the class initialization method (page 178 col. 1 lines 6-11).

15. As per claim 10, Yantchev teaches system where the deadlock analyzer adds and/or removes one or more nodes and/or arcs from the wait for graph (page 179 col. 2 lines 26-30).

16. As per claim 11, Bak teaches system further comprising a semantic analyzer adapted to analyze a semantic type associated with the class initialization method, where the semantic analyzer provides information concerning a desired initialization check time to the initialization method activator (col. 8 lines 22-30, col. 9 lines 35- 46).

17. As per claim 12, Bak teaches system where the semantic type is one of "exact" and "before field initialization" (701,703,705 fig. 9, col. 9 lines 47-62).

18. As per claim 13, Bak teaches system further comprising a domain uniqueness analyzer adapted to analyze the uniqueness of one or more domains with which the class initialization method and/or the class will interact, where the uniqueness analyzer provides information concerning a desired initialization check time to the initialization method activator (701,703,705 fig. 9, col. 9 lines 47-62).

19. As per claim 14, Bak teaches system where the domain uniqueness is one of "normal" and "domain neutral" (col. 9 lines 55 – 62)

20. Claim 15 is rejected based on the same rejection as claim 1 above.

21. As per claim 16, Bak teaches a computer readable medium containing computer executable components of a system for mitigating problems associated with automatic execution of initialization code (col. 8 lines 22-30, col. 9 lines 35- 46), the system comprising:

a semantic analyzing component adapted to determine a semantic type associated with the initialization method (701,703,705 fig. 9, col. 9 lines 47-62);

a domain uniqueness analyzing component adapted to determine a uniqueness type associated with one or more application domains with which the class will interact(col. 9 lines 55 – 62);

an initialization method activating component adapted to call the initialization method at a pre-determined execution point, where the pre- determine execution point depends on, at least in part, the semantic type and the domain uniqueness(col. 8 lines 22-30, col. 9 lines 35- 46).

22. Yantchev teaches a deadlock analyzing component adapted to determine whether calling the initialization method will create a deadlock, the deadlock analyzing

component further adapted to resolve a deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

23. As per claim 17, Bak teaches a method for mitigating problems associated with automatic execution of class initialization code, the method comprising:
determining whether a class has an initializing method, determining when the initializing method should be run (col. 5 lines 25-34);
associating initialization check code with one or more components associated with a runtime, the check code operable to determine whether a class is initialized (col. 8 lines 23-30);
calling the class initializing method(col. 5 lines 25-34).

24. Yantchev teaches determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

25. As per claim 18, Bak teaches a method where determining when the initializing method should be run comprises: analyzing semantic information associated with the initializing method (701,703,705 fig. 9, col. 9 lines 47-62).

26. As per claim 19, Bak teaches a method where the semantic information comprises an identifier that identifies whether the initializing method desires "exact" or "before field initialization" behavior (701,703,705 fig. 9, col. 9 lines 47-62).

27. As per claim 20, Bak teaches a method where determining when the initializing method should be run further comprises analyzing domain uniqueness information associated with one or more domains with which the class initialization code will interact (701,703,705 fig. 9, col. 9 lines 47-62).

28. As per claim 21, Bak teaches a method where the domain uniqueness information comprises an identifier that identifies whether the initializing method is associated with a "normal" or a "domain neutral environment (col. 9 lines 55 – 62).

29. As per claim 22, Yantchev teaches method where determining whether calling the initializing method will generate a deadlock comprises:

attempting to acquire an initialization lock associated with the class to be initialized, and if the initialization lock cannot be acquired (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61).

identifying a holding thread that is holding the initialization lock (page 178 col. 1 lines 6-11);

locating a node associated with the holding thread, where the node is located in a wait for graph, and analyzing the wait for graph to determining whether a deadlock exists (page 179 col. 2 lines 26-30).

30. As per claim 23, Yantchev teaches method where analyzing the wait for graph to determine whether a deadlock exists comprises traversing the wait for graph

starting at the node associated with the holding thread and determining whether a cycle is detected in the wait for graph (page 179 col. 2 lines 26-30, page 178 col. 1 lines 6-11).

31. As per claim 24, Yantchev teaches method where resolving the deadlock comprises:

acquiring a lock associated with the wait for graph (page 179 col. 2 lines 26-30);

if a detecting thread that identifies the deadlock previously added one or more arcs and/or nodes to the wait for graph, removing the one or more arcs and/or nodes and the wait for graph(page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61);

releasing the lock associated with the wait for graph and the detecting thread interacting with the class as though the class was initialized (page 179 col. 2 lines 26-30).

32. As per claim 25, Yantchev teaches method where determining whether calling the initializing method will generate a deadlock comprises:

an acquiring thread attempting to acquire an initialization lock associated with the class to be initialized, and, if the lock can not be acquired, determining whether there is a thread waiting for the acquiring thread to complete its processing and release the initialization lock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61).

33. As per claim 26, Yantchev teaches method where, where resolving the deadlock comprises: if it was determined that there was a tread waiting for the acquiring thread to complete its processing and release the initialization lock, then remaining and interacting with the partially initialized state of the class as though the class were initialized, otherwise blocking until the class becomes initialized (page 178 col. 1 lines 6-11, page 178 col. 2 lines 26-35, page 179 col. 2 lines 50-61).

34. Claim 27 is rejected based on the same rejection as claim 17 above.

35. As per claim 28, Bak teaches a method for mitigating problems associated with automatic execution of class initialization code, the method comprising:
determining whether a class has an initializing method, determining when the initializing method should be run (col. 5 lines 25-34);
analyzing semantic information associated with the initializing method, where the semantic information comprises an identifier that identities whether the initializing method desires "exact" or "before field initialization" behavior (701,703,705 fig. 9, col. 9 lines 47-62);
analyzing domain uniqueness information associated with one or more domains with which the initializing method will interact, where the domain uniqueness information comprises an identifier that identities whether the initializing method is operating in a "normal" or a "domain neutral" environment (col. 9 lines 55-62);

associating initialization check code with one or more components associated with a runtime, the check code operable to determine whether a class is initialized (col. 8 lines 23-30);

calling the class initializing method(col. 5 lines 25-34).

36. Yantchev teaches determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

37. As per claim 29, Yantchev teaches method where determining whether calling the initializing method will generate a deadlock comprises:

attempting to acquire an initialization lock associated with the class to be initialized, and if the initialization lock cannot be acquired (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61);

identifying a holding thread that is holding the initialization lock (page 178 col. 1 lines 6-11);

locating a node associated with the holding thread, where the node is located in a wait for graph, and analyzing the wait for graph to determining whether a deadlock exists (page 179 col. 2 lines 26-30).

traversing the wait for graph starting at the node associated with the holding thread and determining whether a cycle is detected in the wait for graph (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61).

38. As per claim 30, Yantchev teaches method where resolving the deadlock comprises:

acquiring a lock associated with the wait for graph (page 179 col. 2 lines 26-30);

if a detecting thread that identifies the deadlock previously added one or more arcs and/or nodes to the wait for graph, removing the one or more arcs and/or nodes and the wait for graph(page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61);

in the thread that detected that it could not initialize the class because a deadlock existed with another thread that was initializing the class, interacting with the class as though it was initialized (page 179 col. 2 lines 26-30).

39. Claim 31 is rejected based on the same rejection as claim 28 above.

40. As per claim 32, Bak teaches a system for mitigating problems associated with automatic execution of class initialization code, the method comprising:

means for identifying a constructor associated with a class and means for scheduling the running of the constructor (col. 5 lines 25-34);

means for adding code into one or more components generated by a runtime, the code operable to identify whether a class is initialized (col. 5 lines 25-34); and

means for invoking a constructor(col. 5 lines 25-34).

41. Yantchev teaches means for detecting deadlocks between constructors and means for resolving deadlocks between constructors (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

42. As per claim 33, Bak teaches a data packet adapted to be transmitted between two or more components (col. 4 lines 40-45).

43. Yantchev teaches information associated with one or more nodes associated with a wait for graph, where the nodes model one or more threads to be analyzed to determine whether class initialization code will generate a deadlock(page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34); and information associated with one or more arcs associated with a wait for graph, where the arcs model one or more wait for relationships between one or more of the nodes (page 179 col. 2 lines 26-30).

44. As per claim 34, Bak teaches a data packet adapted to be transmitted between two or more components(col. 4 lines 40-45)., the data packet comprising:
a first field adapted to hold information concerning the identity of a thread that is attempting to initialize a class(col. 5 lines 25-34);
a second field adapted to hold information concerning the identity of one or more threads that are waiting for a class to be initialized (col. 9 lines 30-32); and
a third field adapted to hold information concerning the initialization status of a class(col. 9 lines 30-32, col. 8lines 22-30).

45. Yantchev teaches determining whether calling the initializing method will generate a deadlock and if calling the initializing method will generate a deadlock, resolving the deadlock (page 178 col. 1 lines 6-11, page 179 col. 2 lines 50-61, page 178 col. 2 lines 26-34).

Conclusion

46. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nilesh Shah whose telephone number is (571)272-3771. The examiner can normally be reached on 9-5.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571)272-3756.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Nilesh Shah
Examiner
Art Unit 2127

NS
September 23, 2004

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100